International Journal of Computer Networking,
Wireless and Mobile Communications (IJCNWMC)
ISSN(P): 2250-1568; ISSN(E): 2278-9448
Vol. 4, Issue 2, Apr 2014, 93-100

AN APPROACH FOR GRAPHEME TO PHONEME ALIGNMENT FOR SANSKRIT TTS

CHANDRA SEKHARAM BONDU[1] & RAMAKRISHNA S R[2]

[1]System Analyst, Department of Computer Science, Rashtriya Sanskrit Vidyapeetha, Tirupati, Andhra Pradesh, India

[2]Professor, Department of Computer Science, Sri Venkateswara University, Tirupati, Andhra Pradesh, India

ABSTRACT

Text to Speech Synthesis system requires an important module which will convert graphemes to phonemes. Grapheme to Phoneme Conversion module takes sequence of sentences as input and map to corresponding phonetic signal. These phonetic signals also known as utterances are stored as 'wav' files on the machine. There are various methods of associating or aligning these wave files with graphemes. This paper discusses important issues relating to Grapheme to Phoneme alignment for Sanskrit text written in Dēvanāgari script. Rule based syllable clustering/segmentation was found suitable for the task. Rule base was created using Regular Expression Builder and was hand crafted. Further, an approach for associating this Linguistic Syllable Unit with manually segmented utterance units was experimented. Linguistic Syllable Units were segmented using Rule Base from Sanskrit Texts. Further, These Linguistic Syllable units were re-segmented to be aligned with phonetic utterances i.e phonemes, for integrating with Text to Speech Synthesis System for Sanskrit.

KEYWORDS: Phonetic Rules, Linguistic Syllable Unit, Grapheme, Virāma, Mātra, Halant, Svara, Rule Based Syllable Segment

INTRODUCTION

Sanskrit is considered to be mother of most of the Indian Languages. Sanskrit has a rich collection Phonetic texts viz. Śikṣa, prātisākhya, Aṣṭādhyāyi etc., hereafter referred as Sanskrit Phonetics. These texts enumerate Phonetic rules for syllabification in the form of aphorisms. Sanskrit language maintains orthography of utterances with written symbols. Orthography is the term used to indicate 'what is written is what is uttered'. The Writing System irregularity is main cause of complexity in doing Grapheme to Phoneme Conversion [Tatyana2005]. Writing System irregularity implies that there is no one to one correspondence between 'what is written is what is uttered'. Even though Sanskrit text is represented in majorly in Dēvanāgari Script, it is possible to find the same being represented in regional languages of India. In this paper the discussion is limited to Sanskrit Text represented in Dēvanāgari Script. Pāṇinīya Śikṣa mentions sixty three speech Sounds. Among them twenty-one are vowels and remaining are consonants, the details may be found in the Phonetic rules Subhra Basu Ghosh (2003).

Text to Speech Synthesis (hereafter referred as TTS) and Speech recognition technologies are finding fast growth. TTS for any language requires a tool that associates every letter symbol in the stream of text to a corresponding phone or utterance. This task is carried out by Grapheme to Phoneme Alignment Module. Grapheme to Phoneme Alignment is essential for any orthographic language also, for the cases when the number of letter is greater than that of phonemes of a language [Tatyana2005]. Rule base extracted from Sanskrit Phonetic was used to build this module. The research was carried out in Rashtriya Sanskrit Vidyapeetha, a Central Sanskrit University, where both Sanskrit Linguists and Computer

Professionals are available. This University has rich collection Unicode text digitized. Sanskrit being rarely spoken finds less number of researchers working for it.

Definitions of Important Terminology Relating to Sanskrit Phonetics

The most usual term used in Sanskrit phonology for syllable is *akshara*, which literally means imperishable. In theory a syllable consists of a sequence of sounds containing one peak of prominence. This peak of prominence is known as *svara* or vowel. In practice it is often impossible to define the limits of a syllable, because there is no means of fixing any exact points of minimum prominence Vidhata Mishra, (1972). Sanskrit is a vowel-centric language and hence consonants must be clustered with a vowel preceding it or following it. This process of clustering consonant or consonant cluster with either preceding vowel or next vowel is known as syllabification. This syllabification is governed by set of Sanskrit phonetic rules. As for as Sanskrit Phoneticians, are concerned, they prefer to adhere to rules enumerated in Phonetic Texts. The grapheme cluster that is obtained by associating consonant conjuncts with either vowel preceding or next to it is called Linguistic Syllable Unit (LSU). LSU should be recombined to align with phonemes.

Different methods were proposed to handle this Grapheme to Phoneme Alignment. In their research work various researchers proposed decision based tree Alan W Black, et al (1998), Ananlada (2000), statistically based Stanley (2003), and pronunciation analogy Example Based Grapheme phoneme conversion as proposed by Paisarn et al (2006) segmented the syllables either by rule based procedure or by statistically based syllable segmenter. Paisarn proposed this for Thai Language. As the rules used by the paisam did not coincide with the phonetic rules of Sanskrit, it was felt to develop Rule Based syllable segmenter for Sanskrit. In This paper descripes Grapheme to Phoneme Alignment with Rule Based Segmentation

GRAPHEME TO PHONEME ALIGNMENT

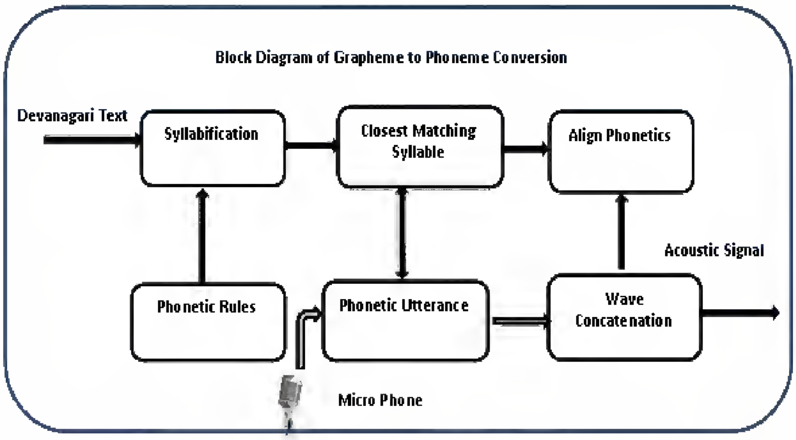

Figure 1: Block Diagram of Grapheme to Phoneme Conversion

The Following steps were taken up in sequence to take up grapheme to phoneme alignment task for Sanskrit text.

Record all phonemes possible

Manually segment the utterances in groups using Wave lab or SFS, Wave Pad Apps.

Syllabification of stream of Dēvanāgari texts using proposed algorithm (Rule Based Syllable Segmenting)

Divide LSU into Phoneme alignable Units basing utterance data available.

Integrate Graphme to Phoneme module with Text to Speech Synthesis for Sanskrit.

Compare manually

Creating Phoneme Utterance

A Sanskrit Phonetician, with no defects of speech, as enumerated in Phonetic texts was chosen to create samples for this experiment. A good quality recorder, of sony make, PCM-D50, Linear PCM Recorder used to record the signal. The Recording was done in Acoustic Studio in University. Low Cut Filter, Limiter Options were enabled. Sampling rate was set at 44.1 kHz; bit rate was set at 16 bit stereo. The Speech Talent was asked to utter all possible combinations from the basic Sanskrit phone set, giving a gap of silence between each combination. Further, Amara Kosha, a Sanskrit Nighantu was depended in selecting specific consonant conjunct contexts, for recording. This wave files were segmented using WAVELAB 4.0 and were named accordingly. The files generated were so named that Unicode numbers corresponding to that utterance sequence formed the part of wave file name. Character "O" was inserted in between each Unicode number to isolate each phoneme. For example utterance symbolised 'क्ष' is named as "2325O2359", 'श्री' is named as 2358O2352O2368. These numbers are decimal coded Unicode numbers corresponding to क, ष, श, र, इ matra respectively. As can be seen from the algorithm ASW () function returns Decimal Coded value for the Unicode string, in Microsoft environment.

Syllabification

The following are Syllabification Rules extracted from Sanskrit Phonetic Texts: These rules were used in next to char function

- Vowel is the nucleus of the syllable.
- Consonant or consonant cluster should be associated with succeeding vowel
- The final consonant in the word will follow preceding vowel.
- Anusvara and visarga belong to preceding vowel.
- The first letter of consonant conjunct go with either preceding vowel or succeeding vowel
- The kramaja letter which means a duplicated letter is a part of previous vowel.
- According to Taittiriya Pratisakhya, the plosive in the group plosive + spirant belong to succeeding vowel.
- In a clustered consonant conjunct consonant + semivowel, consonant goes with succeeding syllable.
- Permitted finals rule says that only one consonant is allowed after last vowel in the word.
- Virama characters define the boundaries of the words or sentences.
- Halant is a symbol which is used to create boundary between to phones of phone set, which has no utterance mapping.
- White Space, coma, Full Stop, Question marks take 1,2,3,4 matras duration silence. These are known as virama in Sanskrit phonetics.

The code for next to char function is as shown on the figure 2 and declaration on figure 3. The next to char function takes two parameters. First parameter Pd text is Dēvanāgari character string, current _ character _ pos is from where consonant conjunct starts. The function return an integer where the succeeding vowel position in the string exists, to which consonant conjunct should be attached to form a syllable. For example, if next to char function is called with Pd text ="अक्षर" and different current _ char _ pos the following output is obtained. The above string can be obtained with "अ+क+्+ष + र" .Here are five phones in this Dēvanāgari string

```
Public Function nexttochar(pdtext , current_char_pos) As Integer'
' This Function returns an integer indicating number of phonemes that can be cluster to succeeding vowel
'--- FIRST CHAR FROM CURRENT POS IS VOWEL
If Regex.IsMatch(Mid(pdtext, current_char_pos, 1), vowel) Then
 If Regex.IsMatch(Mid(pdtext, current_char_pos + 1, 1), anusarg) Then
        nexttochar = current_char_pos + 1
Else
        nexttochar = current_char_pos
    End If
 End If
    '--- FIRST CHAR FROM CURRENT POS IS CONSONANT
 If Regex.IsMatch(Mid(pdtext, current_char_pos, 1), consonant) Then
    nexttochar = current_char_pos
If Regex.IsMatch(Mid(pdtext, current_char_pos + 1, 1), virama) Then
        nexttochar = current_char_pos
    ElseIf Regex.IsMatch(Mid(pdtext, current_char_pos + 1, 1), anusarg) Then
            nexttochar = current_char_pos + 1
    ElseIf Regex.IsMatch(Mid(pdtext, current_char_pos + 1, 1), matra) Then
        If Regex.IsMatch(Mid(pdtext, current_char_pos + 2, 1), anusarg) Then
          nexttochar = current_char_pos + 2
        Else
          nexttochar = current_char_pos + 1
        End If
    ElseIf Regex.IsMatch(Mid(pdtext, current_char_pos + 1, 1), "?") Then
        nexttochar = nexttochar(pdtext, current_char_pos + 1)
        If Regex.IsMatch(Mid(pdtext, current_char_pos + 2, 1), virama) Then
          nexttochar = current_char_pos + 1
        Else
          nexttochar = nexttochar(pdtext, current_char_pos + 1)
        End If
    End if
ElseIf Regex.IsMatch(Mid(pdtext, current_char_pos, 1), "?") Then
      nexttochar = nexttochar(pdtext, current_char_pos + 1)
ElseIf Regex.IsMatch(Mid(pdtext, current_char_pos, 1), virama) Then
      nexttochar = current_char_pos
End If
Return nexttochar
End Function
```

Figure 2: Algorithm to Find Succeeding Syllable Position to which the Consonant Conjunct from Current _ Char _ Pos can be Attached

```
Declaration for the Algorithm explained.

vowel = "[अआइईउऊऋॠएऐओऔअंअः]"

consonant = "[कखगघचछजझञटठडढणतथदधनपफबभमयरलवशषसह]"

matra = "[ािीुूृॄेैोौ]"

virama = "[,\s ।]"

anusarg = "[ंंः]"
```

Figure 3: Declaration for the Next to Char Algorithm

Current _ Char _ Pos	Next to Char(Pd Text, Current _ Char _ Pos)	Explanation
1	1	First is independent Vowel Character
2	4	Consonant conjuct starts at pos=2 and is attached to vowel /consonant at pos=4
5	1	Consonant starts and ends at pos 5

Figure 4: Output of Next to Char Function for Various Current _ Char _ Pos and Pd text==”अक्षर”

Linguistic Syllable Unit to Phoneme

The Linguistic Syllable Unit is aligned by manually constructing the mapping table which tries all possible alignments of letter and phones. This is the technique used by Alan et.al (1998).

- The purposes of aligning this LSU with phoneme utterances are
- To generate synthetically a pronunciation of unseen syllable from the available utterances.
- To generate grapheme phoneme alignments
- To construct statistics of synthetically generated phoneme utterances.

```
Function GRAPHEM2PHONEME(LSU) AS STRING
Ui=ASW(Mid(LSU,i,1))
Select Case
Case 1  Len(LSU)=1   Return   U1
Case 2  Len(LSU)=2   if exist(u1ou2) return u1ou2 else (u1+u2 )
Case 3    if exist(u1ou2ou3)  return u1ou2ou3
                  Else  if exist(u1ou2) return (u1ou2 +u3)
                  Elseif exist(u2ou3) return(u1+u2ou3)
                  Elseif not exist(u1ou2) and not exist( u2ou3) return( u1+u2+u3)
Case 4  if exist(u1ou2ou3ou4)  return u1ou2ou3ou4
        Else if exist (u1ou2ou3) return (u1ou2ou3+u4)
        Elseif exist (u2ou3ou4) return (u1+u2ou3ou4)
        Elseif exist(u1ou2)  and exist(u2ou3) return (u1ou2 +u3ou4)
        Elseifexist(u1ou2)  and  not exist(u2Ou3) return (u1 ou2 +u3 +u4)
        Elseif not exist(u1ou2) and exist(u2ou3) return (u1+u2 +u3ou4)
        Elseif not exist(u1ou2) and not exist(u2ou3) return (u1+u2+u3 +u4)
End Select
END FUNCTION
```

Figure 5: Algorithm for Finding Closest Matching Phoneme Sequence

As explained, while automatically naming the utterances recorded a character “o” is inserted in between

corresponding decimal Unicode numbers For example file name corresponding to "अ' is 2309. Similarly, filename corresponding to "ज्ञ" = "ज + ञ" is 2332o2334. Character 'o' between Unicode indicates it is original utterance. It can also be represented as 2332+2334 where + between Unicodes indicates it is not original utterance, but synthesized. Thus, using above algorithm one can map to the signal files available on the system with closest matching syllable. The above algorithm segments each linguistic Syllable unit into possible sequences, while matching the utterance signals available with system. The closest matching syllable can be selected by counting number 'o' characters in the Unicode based file name discussed above. If number of "o" s is more than the number of "+"s then the grapheme to phoneme alignment can be understood closest matching grapheme to phoneme. **"श्र्य"** for example, is Linguistic syllable unit then the above function can return following values

श्+ र् +य	If exist(u1ou2ou3)	2358o2352o2351	Number of o's =2 +'s =0
श्+ र् +य	Else if exist(u1ou2)	2358o2352 +2351	Number of o's=1 +'s=1
श्+ र् +य	Else ifexist(u2ou3)	2358+2351o2351	Number o's=1 +'s=1
श्+ र् +य	Elseif not exist(u1ou2) and not exist(u2ou3)	2358+2352 +2351	Number o's=o +'s=2

Figure 6: Output of Grapheme Phoneme Function

RESULTS

The utterances created included isolated basic phone set with pattern C, V, CM, CA, CMA and VCH. Other triphonic utterances are also recorded using the words listed in Sanskrit dictionary. Linguistic Syllable Unit was segmented using Closest matching Syllable algorithm explained. Because basic phoneset was available, grapheme to phoneme alignment was possible atleast with maximum editing distance, if not with closest matching syllable. The Unicode data available with the University is used for testing with Syllabification and LSU to Phoneme Alignment module. The output of the above modules were manually got verified with Sanskrit linguists. The results showed cent percent accuracy. It was felt that this result could be because, syllabification is done purely on the rule base, whereas other researchers depended on the statistical and data oriented methods.

CONCLUSIONS

From the experiment results Rule Based Segmentation Technique for syllabification of Unicoded Sanskrit Text and Closest Matching Algorithm produced impressive results. These two modules could successfully be integrated with TTS for Sanskrit. These algorithms were experimented with semi automatically segmented phonemes. It was thought the alignment of Linguistic syllable Unit should be performed on automatically segmented phoneme utterances. This rule based algorithm can be successfully tried on any syllable centric orthographic Indian languages.

ACKNOWLEDGEMENTS

The author thanks to Prof. Sairam for giving his voice, Mr. Kamesh, Mr. ChandraSekhar Vedam, Mr. Srinivas for editing the audio sample files, Dr. Dakshinamurthy for providing Acoutic Room, and Mrs. Suvarchala for all her support.

REFERENCES

1. Tatyana Polyakova et.al (2005),"Main Issues in Grapheme-To-Phoneme Conversion For TTS", Procesamiento del Lenguaje Natural, num.35 (2005), pp.29-34
2. Subhra Basu Ghosh(2003), "Sanskrit Phonetics in the Light of Modern Phonology", Abhoy Burmon 91-92.
3. Vidhata Mishra, (1972), "A Critical Study of Sanskrit Phonetics", The Chowkhamba Sanskrit Series Office Varanasi pp
4. Alan W Black, et al (1998) "Issues in Building General Letter to Sound Rules" The 3rd ESCA Workshop on Speech Synthesis, Jenolan Caves, Australia,
5. Ananlada Chotimongkol et.al (2000) "Statistically trained orthographic to sound Models for Thai" In Proceedings of ICSLP 2000
6. Stanley F. Chen (2003), "Conditional and Joint Models for Grapheme to Phoneme Conversion", The 8th Euro speech Geneva, Switzerland
7. Paisarn Charoenpoornswat et.al (2006), "Example Based Grapheme-to-Phoneme Conversion for Thai", Inter Speech 2006-ICSLP pp, 1268-1271